COMPLEXITY OF PATH COVERING PROBLEMS IN ACYCLIC ALTERNATE GRAPHS(Mathematical Foundations of Computer Science and Their Applications)

AUTHOR(S):

UEMURA, Kenji; YAKU, Takeo

数理解析研究所講究録
556巻 1985年 240-249

COMPLEXITY OF PATH COVERING PROBLEMS IN ACYCLIC ALTERNATE GRAPHS

Kenji UEMURA (植村 憲治) Tsuru Univ.

Takeo YAKU (夜久 竹夫) Tokai Univ.

1. Introduction.

There have been many studies about covering of graphs by Yaku-Iwata(1975),Boesch-Gimpel (1977),Iwata(1978)[1] and Yaku(1979)[2]. In [2] an algorithm of finding a branch minimal spanning out forest is given. In this paper we relate the approximate time complexiy of this algorithm within the case of acyclic alternate graphs of finding a maximal path cover.

Definition. Let G=(V,E) be a digraph. Edges (u,v) and (w,x) in E are alternately adjacent, denoted by (u,v)$\sim$(w,x), if they are distinct and u=w or v=x.

The symbol $\overset{*}{\sim}$ denotes the reflexive and transitive closure of $\sim$. Edges e and e′ are alternately equivalent if $e \overset{*}{\sim} e'$. A graph G=(V,E) is alternate if $e \overset{*}{\sim} e'$ for any edges e and e′.

We assume that our alternate graph satisfies next two conditions.

(1) The indegree of any node is no more than 3 and the outdegree of any node is no more than 2.

(2) Graphs are acyclic even if we neglect the direction of arrow and the outdegree of each terminal node is one.

We also assume that there is one special node of outdegree ≠0 called a root and the graph is called a rooted alternate graph.

242 We will evaluate the approximate time complexity, assuming each non-isomorphic rooted alternate graph to appear under the same probability.

2. Maximal path cover.

Let $G_1=(V_1,E_1)$, $G_2=(V_2,E_2),\ldots,G_n=(V_n,E_n)$ be subgraphs of a digraph $G=(V,E)$. If $V_1 \cup V_2 \cup \ldots \cup V_n=V$ and $V_i \cap V_j \neq \phi (i \neq j)$, then $P=\{G_1,G_2,\ldots,G_n\}$ is a <u>partition</u> of G. Each subgraph is called a <u>component</u> (block). The partition P is a <u>path cover</u> if each G_i $(i \leq n)$ satisfies either

(i) G_i is connected and there is a vertex disjoint path $e_1 e_2 \cdots e_l$ $(l \geq 1)$ in G_i such that $\{e_1,e_2,\ldots e_l\}=E_i$, or

(ii) G is a point, that is, $G_i=(\{v\},\phi)$ $(v \in V)$.

For a path cover $P=\{G_1,G_2,\ldots,G_n\}$ of a graph $G=(V,E)$, G(P) denotes the graph

$$G(P)=(G_1 \cup G_2 \cup \ldots \cup G_n).$$

The path cover P is <u>maximal</u> if, for any path cover Q,

$$\#(E-E(P)) \leq \#E-E(Q)), \text{that is }, \#E(P) \geq \#E(Q).$$

3. Algorithms of finding a maximal path cover.

```
     PROCEDURE ALTSEARCH(v,G,F)

/*     G ( input  ) an alternate dag; vertices in G are     */
/*       ( output ) marked 'UNCOVERED','COVERED',           */
/*                  'UNVISITED' and/or 'VISITED'.           */
/*  v (input)  a vertex in G; searching  start from v.      */
/*  F (output) the edges of a dag: a maximal path cover     */
/*                  for G.                                  */

begin

     mark v 'VISITED';

     while vertex u remaining in OUTLIST (v)

     marked 'UNCOVERED' do

     begin

          add(v,u) to F ; mark u 'COVERED';

          while vertex x remaining in INLIST(u)

               marked 'UNVISITED' do

               ALTSEARCH(x,G,F)

     end

end
```

```
PROCEDURE ALTMATCH(G,F)
/*  G (input)  an alternate dag with outdegree          */
/*             atmost two.                              */
/*  F (output) the edges of a maximal path cover        */
/*             for G.                                   */
begin
      choose an arbitray vertex v  in G;
      F:= ϕ ;
      mark all vertices v with outdegree(v)≥1 "UNVISITED";
      find a vertex v with outdegree(v)=1 on a semipath
      from v ;
      ALTSEARCH(v,G,F)
end
```

4. Mean depth of alternate graphs.

Definition. The mean depth of any rooted alternate graph G is defined by

$\sum_{a:\text{terminal node of } G} p(a)*q(a)$ where $p(a)=1/2^K$, and k is the number of branches in the semipath from root to a, q(a) is the length of this semipath.

The mean depth of n-node alternate graphs is defined by WAL(n)/ALT(n), where ALT(n) is number of all n-node rooted alternate graphs satisfying conditions (1)and (2), and WAL(n) is the total sum of mean depths of these n-node alternate graphs.

Proposition. Let TR(n) be the number of n-node alternate graphs whose roots are terminal nodes and WTR(n) be the total sum of mean depths of these n-node alternate graphs. Then next equations hold.

$$(3)\quad TR(2n)=\sum_{i=0}^{n-2}[TR(2n-2-i)*TR(i)]+TR(n-1)*(TR(n-1)+1)/2 \qquad (n\geq 2)$$

$$TR(2n+1)=\sum_{i=0}^{n-1}TR(2n-1-i)*TR(i) \qquad (n\geq 1)$$

$$TR(0)=TR(1)=1,TR(2)=0$$

$$(4)\quad ALT(2n)=\sum_{i=1}^{n} TR(2n+1-i)*TR(i)$$

$$ALT(2n+1)=\sum_{i=1}^{n}[TR(2n+2-i)*TR(i)]+TR(n+1)*(TR(n+1)+1)/2$$

$$+TR(n+1)*(TR(n+1)+1)/2$$

$$(5)\quad WTR(2n)=WTR(2n-2)+(\sum_{i=1}^{2n-3} WTR(2n-2-i)*TR(i)+WTR(n-1))/2$$

$$+2*TR(2n) \qquad (n \geq 1)$$

$$WTR(2n+1)=WTR(2n-1)+\sum_{i=1}^{2n-2} WTR(2n-1-i)*TR(i)/2+2*TR(2n+1)$$

$$(n \geq 1)$$

$$WTR(0)=WTR(1)=0$$

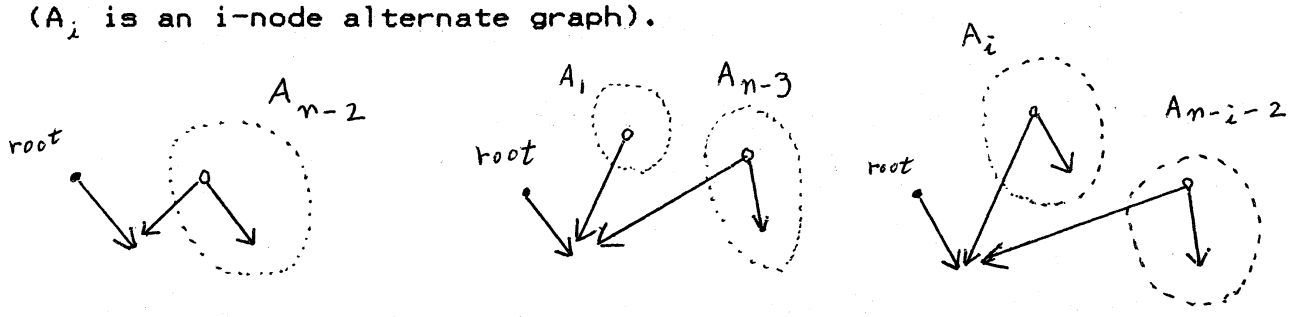
$$(6)\quad WAL(2n)=WTR(2n)+(\sum_{i=2}^{2n-1} WTR(2n+1-i)*TR(i))/2$$

$$WAL(2n+1)=WTR(2n+1)+(\sum_{i=2}^{2n} WTR(2n+2-i)*TR(i)+WTR(n+1))/2$$

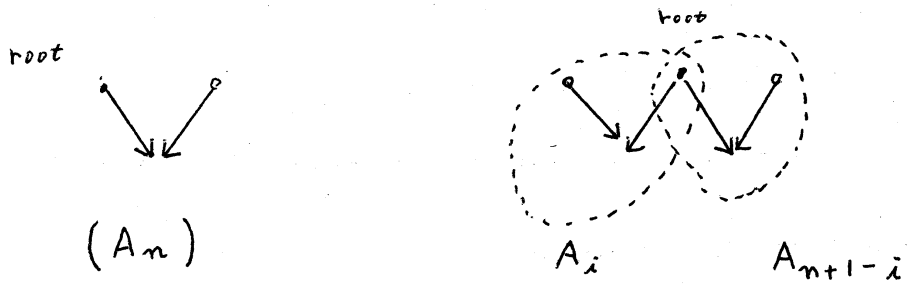
outline of proof.

equation (3). An n-node rooted alternate graph whose root is a terminal node can be uniquely represented below.

(A_i is an i-node alternate graph).



equation (4). An n-node rooted alternate graph can be uniquely represented below.



equation (5). Through unique representations of all n-node rooted alternate graphs whose roots are terminal, each i-node graph appears TR(n-2-i)times except i=n/2-1. And their sum of mean depth is WTR(i)*TR(n-2-i) or WTR(n-2-i)*TR(i).

If i=n/2-1, it appears TR(n/2-1)+1 times and the sum is WTR(n/2-1)*(TR(n/2-1)+1)

And if i≠n-2,there is one more branch for this i-node graph from the root in the associated n-node graph. So their value (except i=n-2) must be divided by two.

Till now each first two edges from the root in any n-node are not counted and their sum is 2TR(n).

equation (6). Almost the same as the case equation (5).

5. Numerical evalutaion.

According to numerical examples below, WAL(n)/ALT(n) seems to increase less than O(log n).

n;	ALT(n);	WAL(n);	WAL(n)/ALT(n)	{WAL(n)/ALT(n)}/log n
20	1746	10307.3	5.90337	1.97059
40	3.77919E+07	2.68012E+08	7.09179	1.92248
60	1.12707E+12	8.56329E+12	7.59785	1.85569
80	3.89366E+16	3.06692E+17	7.8767	1.7975
100	1.46386E+21	1.17888E+22	8.05325	1.74874

These evaluation gives the conjecture that the approximate time complexity of algorithm ALTMATCH on alternate graph of node n satisfying conditions (1)and (2) is bounded by

$$n+\log n$$

References.

(1) S.Iwata, Programs with minimal goto statements, Inform.& Control.37 (1978)

(2) T.Yaku, A linear time algorithm that obtains a maximal matching for graphs, memoire of the Res. Inst. Kyoto Univ. 353(1979),(in Japanese).

(3) M.Ramanath & M.Solomon,Jump minimization in linear time,ACM Trans. on Programing Languages & Systems.6 (1984).